

DRŽAVNO NATJECANJE 2011.

OSNOVNE ŠKOLE BASIC/PASCAL/C/C++

OPISI RJEŠENJA ZADATAKA

Ovo su opisi algoritama kojima su se mogli riješiti navedeni zadaci. Direktne implementacije se nalaze u priloženim kodovima.

1. zadatak (5. razred)	TTT	40/200 bodova
-------------------------------	------------	----------------------

U zadatku se garantira pobjeda igrača koji je postavljao križiće. Za riješiti ovaj zadatak treba provjeriti svih osam mogućnosti na koje je bilo moguće postaviti 3 uzastopna križića.

Pseudo kod:

```
//vrijednost varijable tttij je znak u i-tom retku i j-tom stupcu  
ako je (ttt11='X') i (ttt12='X') i (ttt13='X') tada  
    ispiši('REDAK 1');
```

I slično tako za preostalih sedam mogućnosti.

2. zadatak (5. razred)	PEKARA	80/200 bodova
-------------------------------	---------------	----------------------

U rješenju ovog zadatka treba implementirati postavljene uvjete iz priče.

Pseudo kod:

```
ako je (tom_misli=torba) i (brad_misli=torba) tada  
    ako je tom_vrijeme<brad_vrijeme tada  
        povećaj(tom_krafne,1)  
    inače  
        povećaj(brad_krafne,1)  
inače  
    ako je (tom_misli=torba) tada  
        povećaj(tom_krafne,1)  
    inače  
        ako je (brad_misli=torba) tada  
            povećaj(brad_krafne,1);
```

3. zadatak (5. razred)	TALI5	80/200 bodova
-------------------------------	--------------	----------------------

Prvo moramo provjeriti je li neki od igrača dobio svaki od mogućih brojeva točno jednom. Ako je više igrača dobilo tu kombinaciju, tada je pobjednik onaj čija je oznaka prije po abecedi (što postizemo pametnim korištenjem naredbi). Prilikom implementacije nije potrebno korištenje nizova, ali radi lakšeg kodiranja preporučujemo korištenje navedene strukture.

Pseudo kod (jedan od mogućih):

```
// a[i]-vrijednost koju je igrač A dobio na i-toj kosti  
pobjednik:=-1;  
ako je (a[1]=6) i (a[2]=4) i (a[3]=3) i (a[4]=1) tada  
    pobjednik:=1  
inače  
    ako je (b[1]=6) i (b[2]=4) i (b[3]=3) i (b[4]=1) tada  
        pobjednik:=2  
    inače  
        ako je (c[1]=6) i (c[2]=4) i (c[3]=3) i (c[4]=1) tada  
            pobjednik:=3;
```

U varijabli „*pobjednik*“ pratimo numeričku oznaku pobjednika koju ćemo na kraju lako pretvoriti u slovnu oznaku. Ako pobjednik nije odabran gore navedenim načinom, tada treba tražiti tko je ostvario maksimalni zbroj dobivenih brojeva

Pseudo kod:

```
ako je pobjednik = -1 tada
{
    max:=z[1]; tko:=1;
    za i:=2 do 3 radi
        ako je z[i]>max tada
        {
            max:=z[i];
            tko:=i;
        };
    pobjednik:=tko;
};
```

Nakon ovoga, u varijabli „pobjednik“ imamo brojčanu oznaku pobjednika koju sada lako pretvorimo u slovnu oznaku.

Pseudo kod:

```
ako je pobjednik=1 tada ispiši ('A')
inače
    ako je pobjednik=2 tada ispiši ('B')
    inače
        ispiši ('C');
```

Napomena: Prilikom rješavanja zadataka ovog tipa, uvijek težite univerzalnom optimalnom rješenju koje će biti točno za sve test primjere. Ali, ako ne znate napisati takvo rješenje, tada trebate pokušati osmisliti rješenje koje će donijeti što više bodova. Test primjeri u ovakvim zadacima su složeni tako da se mogu dobiti bodovi i ako se neki od uvjeta iz zadatka ne uspije riješiti.

1. zadatak (6./7. razred)	LOTO	50/50/200 bodova
---------------------------	------	------------------

Za riješiti ovaj zadatak je dovoljno kreirati niz znakova koji će na indeksu odabranog broja imati znak „X“ a na ostalim mjestima znak „*“. Sada je samo potrebno pravilno ispisati taj niz.

Pseudo kod:

```
za i:=1 do 39 radi
    listic[i]:='*';
za i:=1 do 7 radi
{
    učitaj(x);
    listic[x]:='X';
};
za i:=1 do 39 radi
{
    ispiši(listic[i]);
    ako je i mod 3=0 tada
        prijeđi_u_novi_redak;
};
```

Prvo moramo provjeriti je li neki od igrača dobio svaki od mogućih brojeva točno jednom. Ako je više igrača dobilo tu kombinaciju, tada je pobjednik onaj čija je oznaka prije po abecedi (što postignemo pametnim korištenjem naredbi). Prilikom implementacije nije potrebno korištenje nizova, ali radi lakšeg kodiranja preporučujemo korištenje navedene strukture.

Pseudo kod (jedan od mogućih):

```
// a[i]-vrijednost koju je igrač A dobio na i-toj kosti
pobjednik:=-1;
ako je (a[1]=6) i (a[2]=4) i (a[3]=3) i (a[4]=1) tada
    pobjednik:=1
inače
    ako je (b[1]=6) i (b[2]=4) i (b[3]=3) i (b[4]=1) tada
        pobjednik:=2
    inače
        ako je (c[1]=6) i (c[2]=4) i (c[3]=3) i (c[4]=1) tada
            pobjednik:=3;
```

U varijabli „*pobjednik*“ pratimo numeričku oznaku pobjednika koju ćemo na kraju lako pretvoriti u slovnu oznaku. Ako pobjednik nije odabran gore navedenim načinom, tada treba tražiti tko je ostvario maksimalni zbroj dobivenih brojeva. Ako više igrača ima isti najveći zbroj, pobjednik je igrač koji je dobio **više većih brojeva**. Provjera ovog uvjeta postaje jednostavna ako od dobivenih brojeva pojedinog igrača (a oni su već poredani po veličini) kreiramo četveroznamenasti broj.

Pseudo kod:

```
ako je pobjednik =-1 tada
{
    max:=z[1]; tko:=1;
    za i:=2 do 3 radi
        ako je z[i]>=max tada
            ako je z[i]>max tada
            {
                max:=z[i];
                tko:=i;
            }
        inače
            ako je broj[i]>broj[tko] tada
            {
                max:=z[i];
                tko:=i;
            };
    pobjednik:=tko;
};
```

Nakon ovoga, u varijabli „*pobjednik*“ imamo brojčanu oznaku pobjednika koju sada lako pretvorimo u slovnu oznaku.

Pseudo kod:

```
ako je pobjednik=1 tada ispiši ('A')
inače
    ako je pobjednik=2 tada ispiši ('B')
    inače
        ispiši ('C');
```

Napomena: Prilikom rješavanja zadataka ovog tipa, uvijek težite univerzalnom optimalnom rješenju koje će biti točno za sve test primjere. Ali, ako ne znate napisati takvo rješenje, tada trebate pokušati osmisliti rješenje koje će donijeti što više bodova. Test primjeri u ovakvim zadacima su složeni tako da se mogu dobiti bodovi i ako se neki od uvjeta iz zadatka ne uspije riješiti.

3. zadatak (6. razred)**BAGGER****90/200 bodova**

Zadatak se sastoji od rješavanja tri manja odvojena zadatka. Ako se u startu odabere dvodimenzionalno polje (matrica) kao struktura u koju ćemo zapisati broj pokušaja pojedinog igrača na pojedinoj rupi (redci opisuju igrače a stupci rupe), ovaj zadatak postaje relativno jednostavan za riješiti. Za riješiti prvi dio zadatka, nije bilo potrebno korištenje matrice.

Q1: Odredi odnos ukupnog broja udaraca igrača s oznakom "1" i ukupnog PAR-a

Za odrediti odgovor na ovo pitanje je dovoljno posebno zbrojiti vrijednosti iz prvog reda ulaza (ukupni PAR) i drugog reda ulaza (ili prvog reda matrica) te odrediti njihov međusobni odnos.

Pseudo kod:

```
// PAR=zbroj elemenata prvog reda, suma=zbroj elemenata drugog reda
ako je suma=PAR tada
    ispiši('PAR')
inače
    ako je suma<PAR tada
        ispiši('PAR-',PAR-suma)
    inače
        ispiši('PAR+',suma-PAR);
```

Q2: Koliko je rupa osvojio igrač s oznakom "M"

Igrač je osvojio rupu ako je broj brojeva u pojedinom stupcu (pri čemu stupac opisuje pokušaje igrača na odgovarajućoj rupi) koji su veći ili jednaki od zadnjeg elementa u tom stupcu jednak nuli.

Q3: Odredi ukupan poredak igrača

Potrebno je biti oprezan jer se sortiraju oznake na osnovu zadanih pripadajućih vrijednosti.

2./3. zadatak (7./8. razred)**HTZ****60/60/200 bodova**

Zadatak se sastoji od rješavanja tri manja odvojena zadatka. Odaberimo dvodimenzionalno polje (matricu) kao strukturu pomoću koje ćemo najlakše sistematizirati dobivene podatke. U matricu **ocjene** ćemo zapisati ocjene koje je pojedini kandidat (redak) dobio prema odgovarajućem kriteriju (stupac). Zatim ćemo u matricu **bodovi** zapisati vrijednosti nula/jedan ovisno je igrač (opisan tim retkom) dobio bod po odgovarajućem kriteriju (opisanom tim stupcem).

Vrijednost u matrici **bodovi** na poziciji *ij* se dobije tako da se pronade koliko je elemenata u j-tom stupcu matrice **ocjene** (osim onog koji odgovara i-tom retku) jednako elementu na poziciji *ij*. Ako je taj broj jednak nuli tada je pripadajuća vrijednost 1 inače je 0.

Ako se ovako postavi zadatak, rješenje Tonijevog problema je zbroj elemenata u prvom redu matrice **bodovi**. Odgovor na Ivanino pitanje je oznaka retka u kome se nalazi najveći zbroj vrijednosti po redcima (naravno, odgovor je i ta vrijednost). Odgovor na Tonijevo pitanje zahtjeva malo više posla a svodi se na sortiranje oznaka kandidata ovisno o zbroju njihovih bodova. Ali, treba biti oprezan u situaciji kada su zbrojevi bodova dvojice kandidata prilikom sortiranja jednaki. Tada treba napraviti dodatnu provjeru jesu li oznake kandidata u dobrom poretku jer zadatak traži „ako više kandidata ima isti broj bodova, prije treba ispisati onoga čija je oznaka prije po abecedi“.

3./2 zadatak (7./8. razred) GOOGLE /FACEBOOK 90/70/200 bodova

Rješenje za 60% bodova je simulacija uvjeta iz zadatka. Dovoljno je pamtit i koliko je vrijeme čekanja na obradu na pojedinoj stanici. Stanica na koju ćemo poslati novi podatak mora imati najmanju tu vrijednost. Naravno, u obzir treba uzeti i protok vremena. Pojašnjenje cijelog zadatka je nešto složenije.

U ovom zadatku je važno pažljivo organizirati podatke, jer se u protivnom rješenje može jako zakomplicirati. Primijetimo da je za svaki stroj s dovoljno pamtit i sljedeće:

- koliko zadataka čeka izvršavanje na stroju s , u polju `stroj_brZadataka[s] =: Ns`;
- oznake zadataka koji čekaju na stroju s , u 2D-polju `stroj_zadatak[s,1] ... stroj_zadatak[s, Ns]`;
- trenutak kada svaki od zadataka koji čekaju na stroju s započinje s izvršavanjem, u 2D-polju `stroj_kadPocinje[s, 1]...stroj_kadPocinje[s, Ns]`;
- trenutak kada stroj s završava s zadnjim od zadataka koji čekaju, u polju `stroj_kadZavrsava[s]`;
- je li stroj s pokvaren.

Pretpostavimo da je z oznaka zadatka koji dolazi u trenutku t . Sada nije teško napisati funkciju koja raspoređuje taj zadatak na stroj.

`staviZadatak(z, t)`:

- za svaki stroj s koji nije pokvaren, neka je $P_s = \max \{t, stroj_kadZavrsava[s]\}$;
- zadatak z će se izvršiti na onom stroju s za kojeg je P_s najmanji;
- za taj stroj:
 - povećamo `stroj_brZadataka[s]`, odnosno N_s za jedan;
 - stavimo `stroj_zadatak[s, Ns] = z`;
 - stavimo `stroj_kadPocinje[s, Ns] = Ps`;
 - stavimo `stroj_kadZavrsava[s] = Ps + trajanje zadatka z`.

Također, lako je napisati funkciju `makniZadatak(s)` koja briše prvi zadatak sa stroja s i vraća njegovu oznaku: potrebno je samo pomaknuti sve elemente u poljima `stroj_zadatak[s,2] ... stroj_zadatak[s,Ns]` i `stroj_kadPocinje[s,2]...stroj_kadPocinje[s, Ns]` za jedno mjesto ulijevo, te za jedan smanjiti `stroj_brZadataka[s]`.

Kada imamo ove dvije funkcije, možemo napisati petlju u glavnom programu koja obavlja cijelu simulaciju:

$t = 1$;

ponavlja:

- ako postoji zadatak z koji dolazi u trenutku t , pozovi funkciju `staviZadatak(z, t)`;
- ako na nekom stroju s postoji zadatak koji započinje s obradom u trenutku t , pozovi funkciju $z = \text{makniZadatak}(s)$; ako je maknut zadnji zadatak, ispiši brojeve s i $t + \text{trajanje tog zadatka}$, te izađi iz petlje;
- ako postoji kvar na nekom stroju s koji se dogodio u trenutku t , onda:
 - označi da je stroj s pokvaren;
 - sve dok je `stroj_brZadataka(s) > 0`
 - $z = \text{makniZadatak}(s)$;
 - `staviZadatak(z, t)`;
- $t = t + 1$;

Uočimo prvo da se najmanje šibica (samo dvije) troši za znamenku 1. Zbog toga, ako je $a=0$, onda se najveći broj kojeg možemo sastaviti sa N šibica sastoji od $N/2$ jedinica (ako je N paran), odnosno, nema smisla promatrati brojeve veće od $(N+1)/2$ (ako je N neparan). Dakle, zadatak u kojem nam nije zadan interval $[a, b]$ možemo svesti na zadatak u kojem je $a=1$, dok se broj b sastoji od $N/2$ ili $(N+1)/2$ jedinica.

Postavimo malo jednostavnije pitanje: koliko ima nizova od točno z znamenki koji se mogu prikazati sa N ili manje šibica? Označimo taj broj sa $B(N, z)$. Možemo napisati rekurzivnu funkciju koja računa $B(N, z)$; ovdje je `sibice[0]...sibice[9]` broj šibica potreban za prikaz pojedine znamenke 0, 1, ..., 9.

postavi sve elemente polja PAMTI[1..100, 1..50] na -1;

B(N, z):

- *ako je $N < 0$, vrati 0;*
- *inače, ako je $z = 0$, vrati 1;*
- *inače, ako je $PAMTI[N, z] \neq -1$, vrati $PAMTI[N, z]$;*
- *inače:*
 - *koliko = 0;*
 - *for znam = 0, 1, ..., 9 (broj znam je znamenka kojom započinjemo niz)*
 - *koliko = koliko + B(N-sibice[znám], z-1);*
 - *PAMTI[N, z] = koliko;*
 - *vrati koliko;*

Koristili smo tehniku **memoizacije** (polje *PAMTI*), kako bi izračunavanje bilo dovoljno brzo, odnosno, kako se jednom izračunati rezultati ne bi opet iznova računali.

Sada možemo izračunati koliko ima brojeva manjih od x koji se mogu prikazati sa N ili manje šibica; označimo taj broj sa $Z(N, x)$:

Z(N, x):

- *ukupno = 0;*
- *neka broj x ima zx znamenki;*
- *for z = 1, 2, ..., zx-1 (brojimo sve z-znamenkaste brojeve manje od x)*
 - *for prva=1, 2, ..., 9(njihova prva znamenka može biti 1, 2, ..., 9)*
 - *ukupno = ukupno + B(N - sibice[prva], z - 1);*
- *sada još treba prebrojiti sve zx-znamenkaste brojeve manje od x:*
 - *dosad = 0;*
 - *for i = 1, 2, ..., zx*
 - *ako je i = 1, onda stavi P = 1; inače stavi P = 0;*
 - *for z = P, P+1, ..., (i-ta znamenka od x - 1)*
 - *ukupno = ukupno + B(N - dosad - sibice[z], zx - i);*
 - *dosad = dosad + sibice[i-ta znamenka od x];*

Na primjer, ako je $x = 324$, onda prva petlja prebroji sve jedno- i dvoznamenkaste brojeve koji se mogu prikazati sa N ili manje šibica. Druga petlja prvo (kada $i=1$) prebroji sve brojeve oblika 1?? i 2??, pa onda (kada $i=2$) sve oblika 30? i 31?, pa na kraju (kada $i=3$) još provjeri mogu li se prikazati 320, 321, 322 i 323.

Broj kojeg tražimo u zadatku je jednak $Z(N,b) - Z(N,a)+B$, pri čemu je $B=1$ ako je broj b može prikazati, a $B=0$ ako se ne može.

Opisano rješenje rješava sve test-primjere, uključujući i onaj u kojem brojevi a i b imaju po 50 znamenki; oni se učitavaju kao stringovi. U ovom zadatku je bilo moguće na razne jednostavnije načine ispravno riješiti neke test-primjere. Na primjer:

- petlja koja provjerava može li se svaki broj između a i b prikazati pomoću N ili manje šibica bila je dovoljno brza za 3 od 10 test-primjera;
- rekurzija koja konstruira sve tražene brojeve (i samo njih) bila je dovoljno brza za 6 od 10 test-primjera;
- postoji bitno jednostavnije rekurzivno rješenje za slučaj $a=0$.